Instalacja MailCow na docker01 od zera

Typ: procedura

Dotyczy: MailCow Dockerized (gałąź master, sprawdzono w dokumentacji 28.09.2026)

Czas: ~60 min

Efekt: działający MailCow na `docker01`, dostępny pod `poczta.zielona.example`, z kompletem rekordów DNS domeny `zielona.example` i ważnym certyfikatem TLS, gotowy do założenia pierwszej skrzynki.

ZAŁOŻENIA SCENARIUSZA

poczta wchodzi do `docker01` przez pojedynczy `fw01` z jednym łączem światłowodowym (`wan1`, 203.0.113.10), a rekordy A, SPF i PTR dotyczą tylko tego adresu. Przy drugim łączu ze Starlinkiem o publicznym, ale zmiennym adresie poczta wychodzi tylko przez światłowód:

Łącza operatora: światłowód, radiolinia, LTE i 5G oraz przełączanie na FortiGate.

Skróty użyte na tej stronie (15)

Skrót	Co znaczy
ACME	Automatic Certificate Management Environment: protokół, którym Traefik albo certbot automatycznie zamawiają i odnawiają certyfikat w Let's Encrypt, bez ręcznego wniosku
DKIM	DomainKeys Identified Mail: cyfrowy podpis wiadomości e-mail
DNS	Domain Name System: książka telefoniczna internetu, zamienia nazwy na adresy IP
HAProxy	High Availability Proxy: program, który rozdziela połączenia między kilka serwerów i omija te, które nie odpowiadają
HTTP	Hypertext Transfer Protocol: protokół stron WWW
IMAP	Internet Message Access Protocol: protokół, którym klient pocztowy pobiera wiadomości z serwera, zostawiając je tam zamiast kasować
IP	Internet Protocol: adres komputera w sieci; w elektryce: stopień ochrony obudowy (Ingress Protection)
PEM	Privacy-Enhanced Mail: tekstowy format certyfikatu zaczynający się od -----BEGIN

Skrót	Co znaczy
POP3	starszy protokół pobierania poczty na dysk lokalny, w odróżnieniu od IMAP
PTR	rekord DNS odwrotny: zamienia adres IP na nazwę
SMTP	Simple Mail Transfer Protocol: protokół wysyłania poczty
SPF	Sender Policy Framework: rekord DNS mówiący, które serwery mogą wysyłać pocztę domeny
TLS	Transport Layer Security: szyfrowanie połączeń, np. w HTTPS
TXT	typ rekordu DNS przechowującego dowolny tekst, wykorzystywany między innymi do publikowania reguł SPF, DKIM i DMARC
WWW	World Wide Web: strony i aplikacje otwierane w przeglądarce po HTTP albo HTTPS

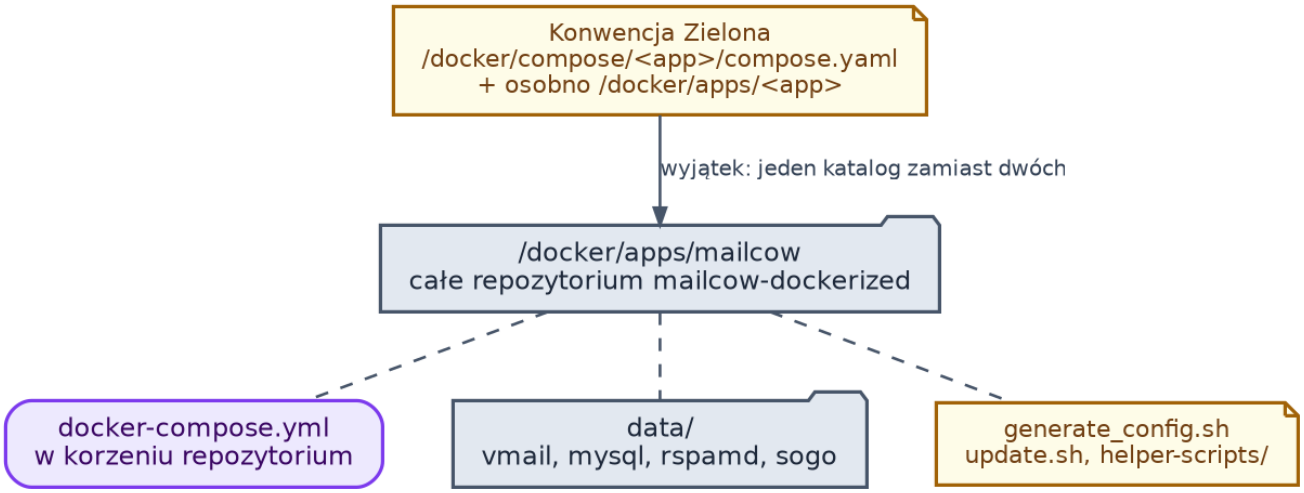
Zanim zaczniemy

> **Z ŻYCIA WZIĘTE**
>
> Tomek dostaje zadanie: „postaw nam MailCow, przechodzimy z Zimbry". Zanim otworzy terminal, szuka gotowego pliku `compose.yaml` do skopiowania, tak jak przy Gitei czy Graylogu. Nie znajduje, bo MailCow działa inaczej: instaluje się z całego repozytorium projektu, nie z jednego pliku.

MailCow, w odróżnieniu od większości aplikacji na `docker01`, nie rozprowadza się jako pojedynczy plik `compose.yaml` do skopiowania i dopasowania. Oficjalny sposób instalacji to sklonowanie całego repozytorium `mailcow-dockerized` i uruchomienie dołączonego do niego skryptu, który sam generuje plik `docker-compose.yml` dopasowany do wykrytego systemu.

> W naszym środowisku dane aplikacji trzymamy w `/docker/apps/<nazwa>`, a plik Compose w `/docker/compose/<nazwa>/compose.yaml`. To konwencja, nie wymóg Dockera: można ją zmienić, byle w całej firmie była jedna. Katalog `/docker` to osobny wolumen `vgDocker/lvDocker`, żeby pełny dysk aplikacji nie zatrzymał systemu.

Uczciwie o wyjątku. Repozytorium `mailcow-dockerized` trzyma w jednym katalogu jednocześnie plik `docker-compose.yml`, skrypty instalacyjne (`generate_config.sh`, `update.sh`, `helper-scripts/`) i katalog `data/` z rzeczywistymi danymi skrzynek, bazy i konfiguracji filtra antyspamowego. Rozdzielenie tego na osobny `/docker/compose/mailcow/compose.yaml` i osobny `/docker/apps/mailcow` złamałoby założenia własnych skryptów MailCow, które oczekują wszystkiego w jednym miejscu. Zamiast walczyć z tym założeniem, traktujemy `/docker/apps/mailcow` jako wyjątek od standardowej konwencji: całe repozytorium, razem z plikiem Compose, ląduje w jednym katalogu na wolumenie `/docker`.



Katalog /docker/apps/mailcow jako wyjątek od konwencji rozdzielenia compose.yaml i danych aplikacji (kliknięcie otwiera pełny rozmiar)

Wymaganie	Uwaga
Docker Engine i wtyczka Compose	sprawdzamy `docker compose version`, wymagane Docker ≥ 24, Compose ≥ 2
Pakiety `git`, `openssl`, `curl`, `jq`	używane przez `generate_config.sh`
Wolne porty `25`, `465`, `587`, `110`, `995`, `143`, `993`, `4190` na hoście oraz `8080` i `8443` na adresie `127.0.0.1`	żaden inny proces na `docker01` nie może ich zajmować; porty `80` i `443` zajmuje Traefik, patrz uwaga niżej
Działający Traefik z sieci `proxy`	opisany na stronie Reverse proxy w Dockerze: Traefik przed wszystkimi aplikacjami na docker01, przed MailCow
Rekord `A` domeny `poczta.zielona.example` wskazujący na publiczny adres Zielonej	zakładamy przed pierwszym uruchomieniem, żeby Traefik od razu mógł wystawić certyfikat

Na `docker01` porty `80` i `443` zajmuje Traefik, a MailCow domyślnie chce tych samych portów (`HTTP_PORT=80`, `HTTPS_PORT=443`). Dwa procesy nie mogą nasłuchiwać na tym samym porcie, więc stosujemy jeden układ: panel WWW i webmail MailCow (kontener `nginx-mailcow`) przenosimy na porty `8080` i `8443` pod adresem `127.0.0.1`, jak zaleca dokumentacja MailCow dla pracy za reverse proxy, a Traefik przekazuje do niego ruch po sieci `proxy`. Protokoły SMTP, IMAP i POP3 to nie HTTP, więc Traefik ich nie obsłuży: porty pocztowe MailCow wystawia dalej sam, bezpośrednio na hoście. To układ dla jednego hosta Dockera. Gdy przed `docker01` i `docker02` stanie para równoważników, MailCow słucha na `0.0.0.0` i dostaje ruch z HAProxy, a certyfikat znów wystawia `acme-mailcow`; opisuje to strona Równoważenie i adres wirtualny: keepalived i HAProxy na parze lb01 i lb02.

Przebieg

Klonowanie repozytorium: git clone

```
sudo mkdir -p /docker/apps
cd /docker/apps
sudo git clone https://github.com/mailcow/mailcow-dockerized mailcow
cd mailcow
```

Repozytorium trafia w całości do `/docker/apps/mailcow`, zgodnie z wyjątkiem opisanym wyżej.

Wygenerowanie konfiguracji: generate_config.sh

```
sudo ./generate_config.sh
```

Skrypt pyta interaktywnie o trzy rzeczy: nazwę hosta poczty (`MAILCOW_HOSTNAME`), strefę czasową i gałąź aktualizacji. Odpowiadamy zgodnie z Zieloną.

Przykład:

```
Mail server hostname (FQDN): poczta.zielona.example
Timezone [Europe/Berlin]: Europe/Warsaw
Please choose a mailcow branch: master
Configuration generated: mailcow.conf
```

Przejrzenie wygenerowanego pliku: mailcow.conf

Skrypt tworzy plik `mailcow.conf` w korzeniu repozytorium. Kilka zmiennych wartych sprawdzenia przed pierwszym uruchomieniem:

```
MAILCOW_HOSTNAME=poczta.zielona.example
MAILCOW_BRANCH=master
TZ=Europe/Warsaw
```

```
HTTP_PORT=8080
HTTPS_PORT=8443
HTTP_BIND=127.0.0.1
HTTPS_BIND=127.0.0.1
DBNAME=mailcow
DBUSER=mailcow
DBPASS=HasloDoBazy
DBROOT=HasloRootaBazy
IPV4_NETWORK=172.22.1
SKIP_LETS_ENCRYPT=y
```

(gdzie `DBPASS` i `DBROOT` to hasła wygenerowane losowo przez skrypt, `IPV4_NETWORK` to wewnętrzna sieć Dockera między kontenerami MailCow, a `SKIP_LETS_ENCRYPT` ustawione na `y` wyłącza własne Let's Encrypt MailCow, bo certyfikatem zajmuje się Traefik, patrz sekcja o certyfikacie niżej)

Generator domyślnie wpisuje `HTTP_PORT=80`, `HTTPS_PORT=443` i `HTTP_BIND=0.0.0.0`, więc cztery pierwsze wartości z listy zmieniamy ręcznie przed pierwszym uruchomieniem (`HTTPS_BIND` dopisujemy, jeśli go brak). Ustawiamy je w `mailcow.conf`, a nie w `docker-compose.yml`, bo tego pliku nie edytujemy. Portów `8081`, `9081` i `65510` nie używamy, bo dokumentacja MailCow wprost tego zabrania.

Podpięcie panelu do Traefika: docker-compose.override.yml

Plik `docker-compose.override.yml` w katalogu `/docker/apps/mailcow` jest przewidziany przez MailCow na własne zmiany i nie jest nadpisywany przez `update.sh`. Dołączamy w nim kontener `nginx-mailcow` do sieci `proxy` i dodajemy etykiety routingu Traefika, tak jak przy każdej innej aplikacji webowej w tej książce:

```
services:
  nginx-mailcow:
    networks:
      proxy: {}
    labels:
      - "traefik.enable=true"
      - "traefik.docker.network=proxy"
      - "traefik.http.routers.poczta.rule=Host(`poczta.zielona.example`)"
      - "traefik.http.routers.poczta.entrypoints=websecure"
      - "traefik.http.routers.poczta.tls=true"
      - "traefik.http.routers.poczta.tls.certresolver=letsencrypt"
```

```
      - "traefik.http.services.poczta.loadbalancer.server.port=8080"

networks:
  proxy:
    external: true
```

Traefik kończy TLS i przekazuje ruch do `nginx-mailcow` zwykłym HTTP na port `8080`. Do potwierdzenia na instalacji testowej: czy `nginx-mailcow` poprawnie obsługuje ruch na porcie HTTP za Traefikiem (nagłówek `X-Forwarded-Proto`) bez pętli przekierowań, oraz czy rekordy `autodiscover` i `autoconfig` domeny `zielona.example` dopisujemy do reguły `Host(...)`. Sprawdzamy to w dokumentacji MailCow, rozdział o reverse proxy.

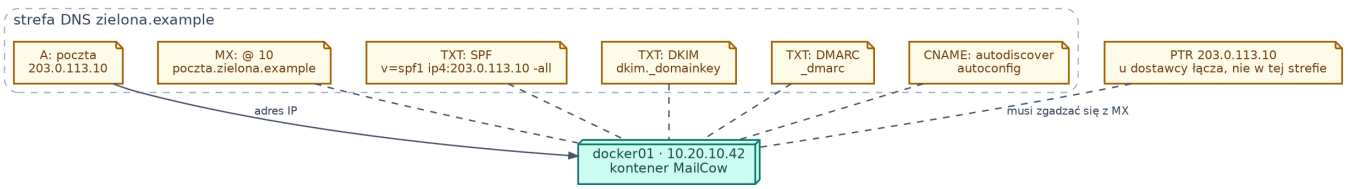
Pobranie obrazów i pierwsze uruchomienie: docker compose

```
cd /docker/apps/mailcow
sudo docker compose pull
sudo docker compose up -d
```

Pierwsze pobranie kilkunastu obrazów potrafi zająć kilka minut, zależnie od łącza.

Rekordy DNS domeny zielona.example

Zakładamy komplet rekordów w strefie `zielona.example`, zanim spróbujemy się zalogować z zewnątrz.



Rekordy DNS potrzebne dla MailCow w strefie zielona.example (kliknięcie otwiera pełny rozmiar)

Rekord	Wartość w Zielonej	Po co
`A`	`poczta` → `203.0.113.10`	adres, pod którym stoi `docker01` od strony internetu

Rekord	Wartość w Zielonej	Po co
`MX`	`@` priorytet `10` → `poczta.zielona.example`	mówi innym serwerom, dokąd wysyłać pocztę dla `zielona.example`
`TXT` (SPF)	`v=spf1 ip4:203.0.113.10 -all`	tylko ten adres IP ma prawo wysyłać pocztę w imieniu `zielona.example`
`TXT` (DKIM)	`dkim._domainkey` → klucz publiczny z panelu MailCow	publikujemy dopiero po pierwszym zalogowaniu, patrz niżej
`TXT` (DMARC)	`_dmarc` → `v=DMARC1; p=none; rua=mailto:raporty-dmarc@zielona.example`	co zrobić z pocztą, która nie przejdzie SPF ani DKIM; zaczynamy od samej obserwacji (`p=none`) i raportów, a skrzynkę na raporty i przejście do `quarantine` i `reject` opisuje strona Droga DMARC od p=none do p=reject z raportami parsedmarc na docker01
`CNAME`	`autodiscover`, `autoconfig` → `poczta.zielona.example`	klienci pocztowi (Outlook, Thunderbird) same znajdują ustawienia serwera
`CNAME`	`mta-sts` → `poczta.zielona.example`	serwer, z którego nadawcy pobierają plik polityki MTA-STS; razem z rekordem TXT `_mta-sts` (`v=STSv1; id=...`), strona MTA-STS i TLS-RPT: szyfrowanie poczty między serwerami, a BIMI informacyjnie
`TXT` (TLS-RPT)	`_smtp._tls` → `v=TLSRPTv1; rua=mailto:raporty-tls@zielona.example`	adres, na który nadawcy raportują kłopoty z szyfrowaniem, strona MTA-STS i TLS-RPT: szyfrowanie poczty między serwerami, a BIMI informacyjnie
`CAA`	`@` → `0 issue "letsencrypt.org"`, `0 issuewild ";"`, `0 iodef "mailto:it@zielona.example"`	tylko Let's Encrypt może wystawić certyfikat dla naszych nazw, także dla `poczta` i `mta-sts`; strona Rekordy publiczne Zielonej, DNSSEC i CAA
`PTR`	`203.0.113.10` → `poczta.zielona.example`	ustawia dostawca łącza Zielonej, nie my w tej strefie, patrz uwaga niżej

> Rekordu PTR nie da się dodać samodzielnie w panelu DNS domeny `zielona.example`: musi go ustawić właściciel puli adresów IP, czyli dostawca łącza internetowego Zielonej. Bez zgodnego PTR znaczna część serwerów pocztowych na świecie odrzuci pocztę z `docker01` niezależnie od poprawnego SPF i DKIM, co opisujemy szerzej na stronie Jak działa poczta: MTA, MDA, MUA, SMTP, kolejka.

Rekord DKIM generuje sam MailCow dopiero po założeniu domeny w panelu, dlatego kolejność jest odwrotna niż przy pozostałych: najpierw logujemy się do panelu i zakładamy domenę `zielona.example`, dopiero potem kopiujemy wygenerowany klucz publiczny do rekordu TXT.

Certyfikat TLS: kontener acme-mailcow

> **Pojęcie: acme-mailcow.** Kontener wbudowany w instalację MailCow, wystawiający i odnawiający certyfikat Let's Encrypt automatycznie, bez ręcznej ingerencji, o ile port `80` jest osiągalny z internetu dla walidacji domeny. Po poprawnym ustawieniu rekordu `A` certyfikat pojawia się zwykle w ciągu kilku minut od pierwszego uruchomienia kontenerów.

> Na `docker01` port `80` należy do Traefika, który ma własny mechanizm Let's Encrypt i przechwytuje żądania walidacyjne `/.well-known/acme-challenge/`. Dlatego `acme-mailcow` wyłączamy (`SKIP_LETS_ENCRYPT=y`), a certyfikat dla `poczta.zielona.example` wystawia Traefik (etykieta `certresolver=letsencrypt` z poprzedniego kroku). Usługi pocztowe (SMTP, IMAP, POP3) nie przechodzą przez Traefika, więc dostają ten sam certyfikat z plików.

Certyfikat dla usług pocztowych: pliki cert.pem i key.pem

Do czasu podmiany MailCow używa certyfikatu samopodpisanego, który klienci pocztowi zgłaszają jako niezaufany. Pełny łańcuch i klucz prywatny wyciągamy z magazynu Traefika (`acme.json`) do plików PEM i kopiujemy do katalogu `data/assets/ssl`:

```
sudo cp /sciezka/do/fullchain.pem /docker/apps/mailcow/data/assets/ssl/cert.pem
sudo cp /sciezka/do/privkey.pem /docker/apps/mailcow/data/assets/ssl/key.pem
cd /docker/apps/mailcow
sudo docker compose restart postfix-mailcow dovecot-mailcow nginx-mailcow
```

Sposób wyciągania certyfikatu z `acme.json` (na przykład osobnym narzędziem, uruchamianym po każdym odnowieniu) do potwierdzenia na instalacji testowej. Po każdym odnowieniu certyfikatu, mniej więcej co 60 dni, pliki trzeba podmienić i powtórzyć restart, najlepiej skryptem. Dokumentacja MailCow, rozdział o reverse proxy, opisuje ten sam schemat z zewnętrznym klientem ACME.

Na hoście bez Traefika (`SKIP_LETS_ENCRYPT=n`, `HTTP_PORT=80`) zachowanie jest inne: `acme-mailcow` robi to wszystko sam i obserwujemy go w logu:

```
sudo docker compose logs -f acme-mailcow
```

Przykład:

```
acme-mailcow  | Certificate for poczta.zielona.example issued successfully.
acme-mailcow  | Reloading nginx-mailcow, postfix-mailcow, dovecot-mailcow
```

Pierwsze logowanie

Otwieramy `https://poczta.zielona.example` i logujemy się domyślnym kontem administratora panelu, które MailCow wypisuje po instalacji w pliku `mailcow.conf` w komentarzu, oraz w logu pierwszego startu kontenera `php-fpm-mailcow`. Po pierwszym logowaniu zmieniamy hasło od razu w zakładce administratora.

◫ ZRZUT EKRANU DO UZUPEŁNIENIA

Zrzut do uzupełnienia: ekran logowania panelu MailCow pod `https://poczta.zielona.example`, z widocznym polem „Username" i „Password" oraz logo MailCow.

Weryfikacja

Sprawdzamy, że wszystkie kontenery wystartowały i są zdrowe:

```
cd /docker/apps/mailcow
sudo docker compose ps
```

Przykład:

```
NAME                                IMAGE                     STATUS
mailcowdockerized-postfix-mailcow-1   ghcr.io/mailcow/postfix   Up 3 minutes (healthy)
mailcowdockerized-dovecot-mailcow-1   ghcr.io/mailcow/dovecot   Up 3 minutes (healthy)
mailcowdockerized-nginx-mailcow-1     ghcr.io/mailcow/nginx     Up 3 minutes (healthy)
mailcowdockerized-acme-mailcow-1      ghcr.io/mailcow/acme      Up 3 minutes (healthy)
```

Sprawdzamy odpowiedź panelu WWW i ważność certyfikatu:

```
curl -Iv https://poczta.zielona.example 2>&1 | grep -E "HTTP|subject"
```

Przykład:

```
* subject: CN=poczta.zielona.example
HTTP/2 200
```

Sprawdzamy poprawność opublikowanych rekordów DNS z zewnętrznego hosta:

```
dig +short MX zielona.example
dig +short TXT zielona.example
dig +short -x 203.0.113.10
```

Na koniec wysyłamy testową wiadomość z nowo założonej skrzynki na adres w zewnętrznej domenie i otwieramy w niej pełne nagłówki: dopiero wpisy `spf=pass`, `dkim=pass` i `dmarc=pass` po stronie odbiorcy potwierdzają, że cały komplet DNS działa poprawnie, nie tylko sam fakt dostarczenia wiadomości.

Wycofanie zmian

Zatrzymanie kontenerów bez kasowania danych skrzynek i bazy:

```
cd /docker/apps/mailcow
sudo docker compose down
```

Pełne usunięcie razem z danymi, tylko gdy naprawdę zaczynamy od zera:

```
sudo docker compose down -v
sudo rm -rf /docker/apps/mailcow
```

Gdy coś nie zadziała

Objaw	Przyczyna	Co robimy

Kontener `nginx-mailcow` restartuje się w pętli albo nie startuje	w `mailcow.conf` zostały `HTTP_PORT=80` i `HTTPS_PORT=443`, które zajmuje Traefik, albo `8080` lub `8443` zajął inny proces	`sudo ss -tlnp \ | grep -E ':80|:443|:8080|:8443'`
`docker compose up -d`	`mailcow.conf`	`HTTP_PORT=8080` `HTTPS_PORT=8443`
`docker compose up -d`	`mailcow.conf`	`HTTP_PORT=80` `HTTPS_PORT=443

Panel pod `https://poczta.zielona.example` zwraca `404 page not found` od Traefika albo `502 Bad Gateway`	`nginx-mailcow` nie jest w sieci `proxy` albo zły port w `loadbalancer.server.port`	sprawdzamy `docker-compose.override.yml` i `docker network inspect proxy`, port ma być równy `HTTP_PORT` z `mailcow.conf`
Klienci pocztowi zgłaszają niezaufany certyfikat, a panel w przeglądarce jest w porządku	Traefik wystawił certyfikat panelu, ale pliki `cert.pem` i `key.pem` dla usług pocztowych nie zostały podmienione	kopiujemy certyfikat do `data/assets/ssl` i restartujemy `postfix-mailcow`, `dovecot-mailcow`, `nginx-mailcow`
Traefik nie wystawia certyfikatu dla `poczta.zielona.example`	rekord `A` jeszcze się nie rozpropagował albo port `80` nieosiągalny z internetu (FortiGate)	`dig +short A poczta.zielona.example` z zewnątrz sieci Zielonej, sprawdzamy regułę NAT na `fw01` i `docker compose logs traefik`
Wiadomości wychodzące trafiają do spamu mimo poprawnego SPF	brak albo błędny rekord PTR dla adresu `203.0.113.10`	zgłaszamy do dostawcy łącza internetowego, PTR nie da się ustawić w naszej strefie DNS
`generate_config.sh` kończy się błędem uprawnień do plików	skrypt uruchomiony bez `sudo` albo katalog `/docker/apps/mailcow` należy do innego użytkownika	uruchamiamy ponownie z `sudo`, sprawdzamy właściciela katalogu `ls -la /docker/apps`

□□ Ćwiczenia

Ćwiczenie 1: kolejność rekordu DKIM

Zakładamy MailCow od zera i chcemy opublikować wszystkie rekordy DNS jednym zamachem, zanim uruchomimy kontenery. Ustalamy, dlaczego to się nie uda dla rekordu DKIM.

Odpowiedź

Klucz publiczny DKIM generuje dopiero sam MailCow, indywidualnie dla każdej założonej domeny, po pierwszym zalogowaniu do panelu administracyjnego. Nie istnieje więc przed uruchomieniem kontenerów i założeniem domeny w interfejsie, w odróżnieniu od pozostałych rekordów (`A`, `MX`, `SPF`, `DMARC`), które można przygotować z góry.

Powiązane: Reverse proxy w Dockerze: Traefik przed wszystkimi aplikacjami na docker01 · MailCow · Jak działa poczta: MTA, MDA, MUA, SMTP, kolejka · Reverse proxy i równoważenie obciążenia · MailCow: konta z Active Directory

Źródła: docs.mailcow.email, rozdziały „Prerequisites" i „Install mailcow" · github.com/mailcow/mailcow-dockerized, `generate_config.sh`

Revision #25
Created 2026-09-28 17:15:34 UTC by Tomasz Konieczny

Updated 2026-10-26 14:20:31 UTC by Tomasz Konieczny